社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

再利用実績を持つコード片の再利用におけるプログラム構造を考慮した再利用候補の提示

大谷　明央†　石原　知也†　肥後　芳樹†　楠本　真二†

† 大阪大学大学院情報科学研究科
E-mail: †{a-ohtani,t-ishihr,higo,kusumoto}@ist.osaka-u.ac.jp

あらまし　ソースコードの再利用を支援する手法の 1 つとして広く知られているコード片検索手法において，再利用実績を用いる手法では，再利用される可能性の高いコード片として，過去に再利用されているコード片をユーザに提示する．しかし，この手法ではプログラム構造を考慮せずに提示を行うため，複文の途中までを提示範囲に含む可能性がある．この場合，再利用において，提示範囲に途中まで含まれた複文の，提示範囲から除かれた部分を追加する，または，複文そのものを取り除くなどの修正をユーザ自身が行う必要が生じる．本研究では，再利用実績に基づくコード片検索手法において，プログラム構造を考慮して調整を行ったコード片をユーザに提示する手法を提案する．提案手法では，コード片をユーザに提示する際に，提示する範囲について，プログラム構造を考慮して拡大あるいは縮小することで調整を行う．実験の結果，提案手法によって再利用に適したコード片を提示できることを確認した．
キーワード　コードクローン，コード片検索，ソースコードの再利用

Suggesting Reusable Code Based on Structural Unit of Programming Language and Past Reuse

Akio OHTANI†, Tomoya ISHIHARA†, Yoshiki HIGO†, and Shinji KUSUMOTO†

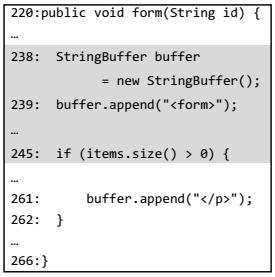
† Graduate School of Information Science and Technology, Osaka University
E-mail: †{a-ohtani,t-ishihr,higo,kusumoto}@ist.osaka-u.ac.jp

Abstract　Code search techniques are well-known as one of the techniques helping code reuse. In particular, code search techniques based on past reuse suggest only code fragments including functionality that users actually want because the techniques suggest only code fragments that have been reused. However, there are cases where the techniques suggest code fragments that include a part of a structural block because the techniques is not based on structural unit of programming languages. In such cases, the users have to manually remove the block, or append the code following the code fragment. In this research, we propose a technique that adjusts the code fragments suggested by the code search technique based on past reuse. The proposed technique expands or contracts the code fragments on the basis of structural unit of programming languages and suggests the code fragments. Therefore, the proposed technique helps users save effort to modify the code fragments in reusing. In this research, we conducted an experiment with 9 participants in order to compare the proposed technique with conventional techniques. As a result, we confirmed that users was able to develop software efficiently by using the proposed technique.
Key words　Code Search, Code Clone, Source Code Reuse

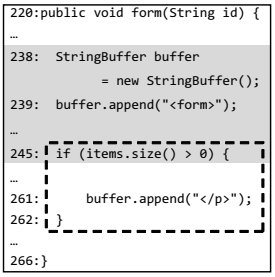
1. はじめに

ソフトウェア開発の効率化を支援するシステムの 1 つにコード片検索システムがある [1] [2] [3]．コード片検索システムは，ユーザが求める機能を表現したクエリを入力すると，そのクエリに関連する機能が実装されているソースコードを出力する．コード片検索システムはユーザに対して複雑な操作を要求しないため，求める機能を実装するコストの削減が期待できる．

しかし，既存のコード片検索システムはプログラム構造のみを基にソースコードを提示するため，ユーザが必要としない機能を含むソースコードを提示することがある．この問題を解決するために，再利用実績に基づくコード片検索が提案されてい

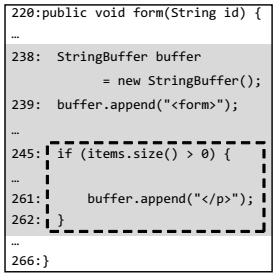


(a) 石原らの手法

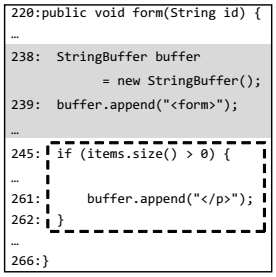


(b) 有益でないコード片の例

図 1　再利用候補の提示と問題点



(a) 分断されていたブロックを全て含む場合



(b) 分断されていたブロックを全て除く場合

図 2　提案手法による範囲の調整

る [4]．この手法を用いたコード検索システムは過去に再利用されたコード片を検出し，その中からユーザの入力するクエリと関連する機能を持つコード片を提示する．過去に再利用されたコード片は今後再利用される可能性が高いと考えられ，このようなコード片を提示することで効率的な再利用支援を行う．

しかしこの手法は，過去の再利用のみを考慮し，プログラム構造については考慮しないため，再利用の観点において必ずしも有益ではない．提示されたコード片がコピーアンドペーストによって再利用された場合，コード片を貼り付けたプログラムにおいて構文エラーが発生する可能性があり，ユーザが構文エラーを修正するコストが発生するためである．このことはソースコードの効率的な再利用を妨げる可能性がある．

そこで本研究では，より効率的な再利用の支援を行うために，再利用実績を持つコード片にプログラム構造を考慮した調整を加えて提示する方法を提案する．つまり，提案手法はプログラムの構造と再利用実績の両方を考慮した提示を行う．また，9 人の被験者の協力のもと実験を行った．実験では，与えられたタスクを被験者が完成させるまでの経過を記録した．被験者はタスクを実装する際に提案手法を含む 3 つのツールを用いている．実験で得た記録を基に，ツールによって提示したコード片と実装したコード片を比較した．その結果，提案手法は既存手法と比べ効率的な再利用の支援を行えることが確認された．

以降，2 章では研究動機，3 章では提案手法とその実装，4 章では実験，5 章では実験結果の考察，6 章では実験の妥当性，7 章では関連研究について述べる．最後に 8 章で本研究のまとめと今後の課題について述べる．

2. 研究の動機

2.1 既存研究

石原らは，過去に再利用されたコード片を検出し，その中からユーザの入力するクエリと関連する機能を持つコード片を提示する手法を提案した [4]．図 1(a) に示すように，石原らの手法では，ソースコード全体を表示し，再利用候補となるコード片の範囲をハイライトすることで提示としている．また，過去の再利用を特定するためにコードクローン検出を利用している．石原らの手法はソースコード解析部とソースコード提示部の 2 つから構成される．ソースコード解析部では，提示対象となるコード片集合を構築するために，対象となるソースコード集合におけるコードクローンの検出と，検出したコードクローンからキーワードの抽出を行い，これらの情報を基にデータベースを作成している．ソースコード提示部では，ソースコード解析部で作成したデータベースを基に，ユーザが入力したクエリに関連するコード片を探し，そのコード片を含むメソッドの重要度，クエリとそのコード片の関連性の強さ，入力として与えられたソースコード集合においてそのコード片が再利用された回数，といった 3 つの指標を基にコード片に順位をつけ，順位にしたがってコード片をユーザに提示する．

2.2 問題点

石原らの手法では過去に再利用されたコード片を検出することで，再利用される可能性が高いコード片を提示している．しかしプログラム構造については考慮しないため，構文的なまとまりを無視してしまう場合がある．このため，構文的なまとまりに従っていないコード片が提示され，そのままコピーされてしまう可能性がある．その場合，コード片を貼り付けたプログラムにおいて構文エラーが発生するため，構文エラーを修正するためのコストがかかる．図 1(b) は石原らの手法によって提示された，再利用の観点において有益ではないコード片の例を示す．図中のハイライトされた部分は石原らの手法によって提示されたコード片を表し，点線で囲まれた部分は構文的なまとまりを示す．この提示において 245 行目から始まる if 文は途中までしか提示範囲に含まれていない．よって，提示されたコード片のみをコピーする場合，ユーザは自分で 246 行目から 262 行目を追加しなければならない．

3. 提案手法

3.1 概要

以降，本論文では，Java を対象として説明し，1 つあるいは複数の文が中括弧で囲まれた箇所を**ブロック**と呼ぶ．石原らの手法では，ブロックを考慮していないため，ブロックを無視した範囲の提示が行われる．前節では，提示範囲にブロックの途中までを含む場合，提示範囲をコピーすることで構文エラーが発生する可能性があることを示した．本研究では，そういった問題を解決するために，石原らの手法で提示される範囲を，ブロックを考慮した範囲に調整する手法を提案する．以降，本論文では，ブロックの途中まであるいは，ブロックの途中からを

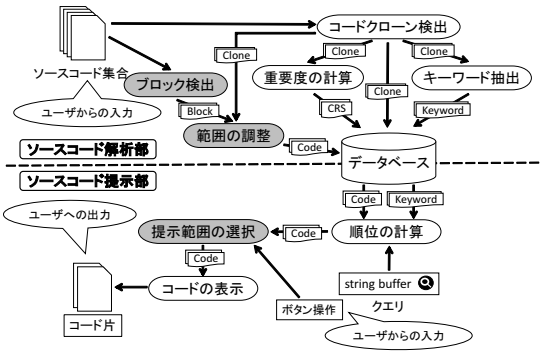


図 3　提案手法の実装

提示範囲に含むことを**ブロックが分断されている**と呼ぶ．提示範囲の調整方法を以下に示す．

1.　提示範囲を拡大し分断されたブロックを提示範囲に含む

2.　提示範囲を縮小し分断されたブロックを提示範囲から除く

ブロックの分断は提示範囲の上端または下端において発生する可能性がある．ここでは，図 1(b) のように石原らの手法による提示範囲の下端でブロックの分断が発生している場合について説明する．図 1(b) において点線で囲まれた 245 行目から 262 行目までの部分が分断されているブロックである．このコード片に対し提案手法を用いて調整した結果の提示範囲を図 2 に示す．分断されていたブロックを提示範囲に含むように拡大した範囲を図 2(a) に，分断されていたブロックを提示範囲から除くように縮小した範囲を図 2(b) に示す．

3.2　提案手法の詳細

本研究では石原らのツールに機能を追加することで提案手法を実装している．実装したツールの概要を図 3 に示す．このツールによるコード検索は以下の 8 ステップで行われる．

STEP1: コードクローン検出

ソースコード集合に含まれるコードクローンを検出する．

STEP2: キーワード抽出

コードクローンに含まれるキーワードを抽出する．

STEP3: 重要度の計算

コードクローンについて，そのコードクローンを含むメソッドの重要度や再利用された回数から重要度を計算する．

STEP4: ブロック検出

ソースコード集合に含まれるブロックを検出する．

STEP5: 範囲の調整

コードクローンとブロックの位置情報から，ブロックが分断されている箇所の範囲を調整しデータベースに保存する．

STEP6: 順位の計算

コードクローンとクエリの関連性の強さや，STEP3 で計算した重要度をもとに順位を計算する．

STEP7: 提示範囲の選択

ユーザからのボタン操作を受けつけ，調整した範囲のうちから提示するものを決定する．

STEP8: コードの表示

提示範囲にハイライトを加えて画面に表示する．

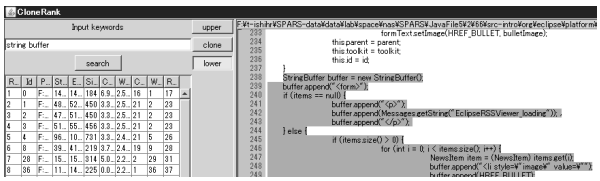


図 4　実装したツールのインターフェース

図 3 に示した処理のうち，色付けされていない処理は石原らの研究での実装を引き継いだ処理である．以降，本研究において実装を行った処理について，実装方法を述べる．

3.2.1　STEP4: ブロック検出

実装したツールでは，Java Development Tools [5] を用いてソースコード中のブロックを検出し，各ブロックについて開始位置と終了位置をデータベースに保存する．

3.2.2　STEP5: 範囲の調整

検出したコードクローンを順に，そのコードクローンが存在するファイル中に含まれる全てのブロックと範囲を比較する．ブロックが分断されていると判定されたものについて，それぞれ 3.1 節で挙げた 2 つの調整方法にあわせて範囲を調整する．全てのコードクローンに対して範囲の調整を行い，拡大後あるいは縮小後の範囲の情報を付与してデータベースに保存する．

3.2.3　STEP7: 提示範囲の選択

実装したツールは，3.2.2 節における調整で得られた範囲のうち，元となったコードクローンとの行数の差が小さい範囲を最初にユーザに提示する．これに対してユーザはボタン操作を行うことで範囲の調整方法を選択できる．図 4 に実装したツールのインターフェースを示し，中央の 3 つのボタンについて，機能を以下に示す．

clone　提案手法による提示範囲と，その元となったコードクローンの範囲を重ねて表示する．

upper，lower　提示範囲の上端あるいは下端について，範囲の調整方法を選択できる．例えば，提示範囲の上端あるいは下端がブロックを含むように拡大されていた場合，ボタンを押すとブロックを除くように縮小した範囲に切り替わる．upper が上端，lower が下端の調整に対応する．

4.　実　　験

提案手法の有効性を評価するために，3. 章で実装したツールを用いて比較実験を行った．

4.1　準　　備

実験では，比較のために以下の 3 つのツールを使用した．

ツール A　コードクローン検出を行わず，メソッド単位でコード片を提示するもの．メソッドの呼び出し関係から計算したメソッドの重要度 [1]，*TF-IDF* 法 [6] を用いて計算したクエリとコード片の関連性の強さを順位付けに反映する．これは既存のコード片検索に対応する．

ツール B　コードクローンを検出し，コードクローン単位でコード片を提示するもの．メソッドの重要度，クエリとコード

表 1　実験で使用したツール

	ツール A	ツール B	ツール C
提示するコード片	メソッド単位	コードクローン単位	コードクローンにブロックを考慮した調整を加えたコード片
メソッドの重要度による順位付け	する	する	する
クエリとの関連性の強さによる順位付け	する	する	する
再利用された回数による順位付け	しない	する	する

片の関連性の強さに加えて，過去に行われた再利用の回数を順位付けに反映する．石原らの手法に対応する．

ツール C　ツール B に手を加え，コード片の提示において，ブロックを考慮して範囲を調整するもの．提案手法に対応する．

ツールの比較を表 1 に示す．B と C の比較から，検索結果の提示におけるブロックの考慮が効率的な再利用を支援するか評価する．A と B または C の比較から，再利用単位でのコード片の提示が有効か評価する．本実験では，既存研究である *SPARS* [1] で使用された，約 400 のプロジェクトで構成され約 19 万の Java ファイルを含むソースコード集合に各ツールを適用し，作成したデータベースを実験に使用した．

4.2 実験方法

本実験では，被験者はいずれかのツールを使用し与えられたタスクを完成させる．被験者はそれぞれ独立にタスクを開始し，表 2 で示したツールを使用してタスクを完成させる．各タスクでは，被験者は入力するクエリを自分で決め，求めるコード片が見つからない場合は，クエリを自由に変更できる．終了条件を満たした時点でタスク終了とする．タスク開始から終了までの被験者の作業状況が画面キャプチャによって録画される．タスクの終了条件は以下の 4 つである．

1. 用意されたユニットテストを通過する．
2. GUI を実装するタスクの場合，実験を監督している者からチェックを受け，合格する．
3. 5 分以上検索しても再利用できるコード片が提示されない．
4. 20 分以上実装してもタスクを完了できる見込みがない．

1 と 2 の場合はタスク完成としてタスク開始から終了までの時間を記録する．3 と 4 の場合はタスク失敗としてその旨を記録する．また，被験者には下記の制限が課される．

- 指定したツール以外の手段でのコード検索を使用しない．
- タスク毎に必ず 1 度は指定したツールで再利用を行う．

4.2.1 被験者

被験者は，大阪大学基礎工学部情報科学科所属の学生 2 人，大阪大学大学院情報科学研究科コンピュータサイエンス専攻所属の修士課程の学生 5 人と博士課程の学生 2 人の計 9 人であり，以下に示すように 3 つのグループに分けられる．

グループ 1　博士課程 1 人，修士課程 1 人，学士課程 1 人
グループ 2　博士課程 1 人，修士課程 1 人，学士課程 1 人

表 2　各グループがタスクを完了させるために使用したツール

	タスク 1,2,3	タスク 4,5,6	タスク 7,8,9
グループ 1	A	B	C
グループ 2	C	A	B
グループ 3	B	C	A

グループ 3　修士課程 3 人

すべての被験者は，Java の経験が半年以上あり，過去に実装した Java プログラムの総行数が 5,000 行以上ある．

4.2.2 タスク

被験者に与えられるタスクの数は 9 であり，その全てが与えられた仕様を満たすメソッドを実装するものである．被験者には宣言部 (修飾子，返り値，名前，引数) だけ記述されたメソッドが与えられ，そのメソッドの満たすべき仕様がコメント部分に記述されている．タスクにはユニットテストまたは，仕様を満たす GUI の例を示した画像が用意されている．ユニットテストが用意されているタスクは，ユニットテストを通過した時点でタスク終了とする．GUI を実装するタスクは，用意された例を確認し，実装が完了したと判断した時点で実験を監督する者のチェックを受け，合格した時点でタスク終了とする．タスク終了の時点での実装のうち，宣言部とコメント部分を除いた実装を，仕様を満たすために被験者が行った実装として記録する．表 3 は本実験で使用したタスクの概要を示している．

4.2.3 評価基準

本実験では，再利用がツールによってどれだけ効率的に支援されているかを，以下の 3 つの指標を用いて判断した．

利用率　必要なコード片のみを提示しているか
貢献率　実装の助けになっているか
所要時間　タスクの完成にどれだけかかったか

利用率はツールによって提示された文の数のうち，コピーして使用された文の数の割合で計算する．貢献率は被験者が実装した文の数のうち，提示したコード片からのコピーによって実装された文の数の割合で計算する．所要時間は被験者がクエリの 1 文字目あるいは実装の 1 文字目を入力した時点から，タスク完成までにかかった時間を用いる．完成しなかったタスクについては，全被験者の全タスクのうち最も時間がかかったものと等しい時間を要したとする．

表 3　実験で使用したタスク

タスク	概要
task1	文字列を分割し整数に変換しソートする
task2	文字列を加工して出力する
task3	swing によりボタンを実装する
task4	ファイルを作成し文字列を書き込む
task5	文字列を分割しソートする
task6	行列の掛け算を行う
task7	テキストファイルを読み込み単語数を数える
task8	swing によりラベルを実装する
task9	排他的論理和の計算を行う

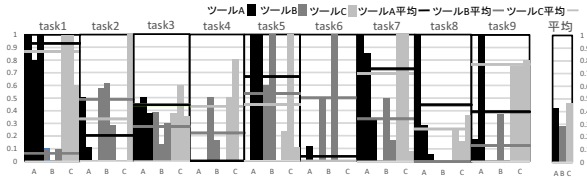


図 5 利 用 率

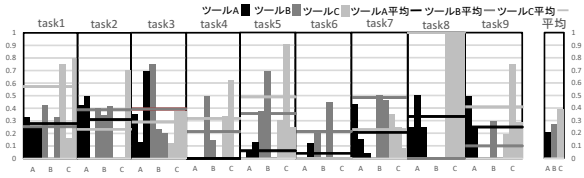


図 6 貢 献 率

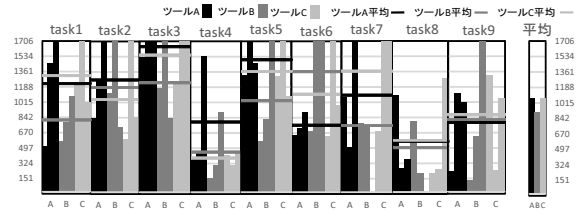


図 7 所 要 時 間

4.3 実 験 結 果

図 5，6，7 はそれぞれ利用率，貢献率，所要時間についてのグラフを示し，縦軸はそれぞれ利用率，貢献率，所要時間を，横軸は被験者がタスクにおいて使用したツールをそれぞれ表している．各タスクにおいて，ツールごとにそれぞれの指標について平均を求めており，グラフ中に横線として示している．また，ツールごとにそれぞれの指標について全体の平均を図右端に示している．利用率は，図 5 より，全体の平均ではツール B が他のツールと比べて低い．タスクごとの平均ではツール A が最も高くなるタスクが多い．貢献率は，図 6 より，全体の平均ではツール C が他のツールと比べて高い．タスクごとの平均ではツール B またはツール C が最も高くなるタスクが多い．所要時間は，図 7 より，全体の平均ではツール B が他のツールと比べて短い．タスクごとの平均でもツール B が最も短くなるタスクが多い．ただし，録画に失敗したため，task7 と task8 におけるツール B のデータが 1 つ欠けており，グラフでは空欄としてある．

5. 考 察

タスクごとに利用率，貢献率を高く，所要時間を短くするツールを調査し表 4 に示した．

5.1 ツール A について

表 4 では，5 つのタスクでツール A の利用率が他のツールと比べて高くなっているが，貢献率は低くなっている．この要因を調査したところ，ツール A での利用率が高いタスクにおいて，ツール A によって提示されたメソッドと，そのメソッドから呼び出される他のメソッドをまとめてコピーしている場面が多く見られた．この場合，提示範囲の全てをコピーしているので，利用率は高くなるが，提示範囲外からも多くコピーしているため，貢献率は低くなる．これは，ツール A が提示したコード片による実装以外に多くの実装が必要になっているということを意味する．ここから，ツール A すなわち既存のコード片検索は，再利用実績に基づくコード片検索と比べて，効率的な再利用を支援できていないといえる．

```
String converted = "";
int spaces = 0;
int ats = 0;
for (int i = 0; i < text.length(); i++) {
  if (text.charAt(i) == ' ') {
    spaces++;
  } else if (text.charAt(i) == '@') {
…
  } else {
    spaces = 0;
    ats = 0;
  }
…
}
```

図 8 被験者が見逃したコード片の例

5.2 ツール B とツール C の比較

表 5 に，ツール B とツール C を比較してタスクごとに利用率，貢献率を高く，所要時間を短くするツールを示した．表 5 より，ツール C の利用率が高いタスクは 6 つあり，貢献率が高いタスクは 5 つある．ここから，提案手法によって提示範囲の調整を行うことで，実装においてより有効なコード片を提示できているといえる．

5.3 ツール C がうまく働かない例

ツール C での利用率と貢献率がともに低いタスクについて調査したところ，再利用できるコード片を被験者が見逃している場面が見られた．図 8 は，被験者が見逃したコード片の例であり，ハイライトされた範囲はツール C によって最初に提示される範囲を，実線で囲まれた範囲は元のコードクローンの範囲を示す．また，点線で囲まれた範囲は，元のコードクローンが分断していたブロックの範囲を示す．この例では，点線内のコード片を再利用することが可能であった．しかし，ツール C は元のコードクローンにより近い範囲として，ブロックを除いた範囲を最初に提示したため，被験者は再利用できるコード片を見

表 4 利用率，貢献率を高く，所要時間を短くするツール

	task1	task2	task3	task4	task5	task6	task7	task8	task9
利用率	A	B	A	C	A	B	A	A	C
貢献率	C	B	B	C	C	B	B	C	C
所要時間	B	C	B	C	B	A	B	B	A

表 5 各タスクにおけるツール B とツール C の比較

	task1	task2	task3	task4	task5	task6	task7	task8	task9
利用率	C	B	C	C	B	B	C	C	C
貢献率	C	B	B	C	C	B	B	C	C
所要時間	B	C	B	C	B	C	B	B	B

逃した．再利用を支援できる場合であっても，最初に提示する範囲が適切でなければ，支援できなくなる．このため，最初に提示される範囲の調整方法は適切に選択されなければならない．

6. 妥当性への脅威

被 験 者

被験者の間で Java プログラミング能力に大きな差がある場合，実験結果に影響を与えることになる．ただし，被験者の学年を考慮してグループに分けたため，グループ間の能力の差はあまり大きくないと考えられる．

タ ス ク

本実験で被験者に与えたタスクは全て小規模な処理を実装するものであった．実験結果からもわかるように，効率的に再利用を支援できるかどうかはタスクに依存する可能性が高いため，タスクの内容を変更することで結果が変わる可能性がある．

データベース

本実験で使用するツールはソースコード集合を解析して作成するデータベースを使用するため，ツールが提示するコード片は，解析するソースコード集合に依存する．

7. 関 連 研 究

Inoue らは，関数の呼び出し関係から各関数の重要度を計算する Component Rank 法と，ソースコード上でキーワードが存在する位置によってキーワードの重要度を計算する Keyword Rank 法を提案し，それらを実装したソースコード検索システムである SPARS を開発した [1]．Component Rank 法では，多くの関数から呼び出される関数や，重要な関数から呼び出される関数の重要度が高くなる．加えて，類似する関数をグループ化し，グループそれぞれに対して，そのグループの重要度として要素の重要度の合計を使用する．Keyword Rank 法では，ソースコードから抽出したキーワードについて，トークンの種類に応じて，そのソースコードの内容を表す上でどれだけの重要度を持っているかを計算する．

McMillan らは，関連する関数やその使用法を知るために，開発者が関数をたどる際のふるまいを表したモデルである Navigation Model と，キーワードの関連性を表したモデルである Association Model を提案し，それらを実装したソースコード検索システムである Portfolio を開発した [3]．Navigation Model では，関数の呼び出し関係を基に各関数の重要度を計算しており，ウェブページの重要度を決定する PageRank 法を応用したものとなっている [7]．Association Model では Spreading Activation 法によってキーワード間の関連性を計算している [8] [9]．

上記の 2 つの手法は関数の呼び出し関係から検索結果の重要度を計算する点で提案手法と類似しており，特に SPARS は類似関数をグループ化する点でも共通している．しかし，これらの手法はプログラム構造のみを基にソースコードを提示する．一方で，提案手法ではコードクローンを基にソースコードを提示するため，ユーザが規模の小さい機能を再利用する際に，ユーザに必要な部分だけを提示するという点で，提案手法は既存手法と比較して，より効率的に再利用を支援するといえる．

8. お わ り に

本研究では，再利用実績を持つコード片の再利用において発生していたコード片への修正のコストを削減するため，プログラム構造を考慮して再利用候補を提示する手法を提案した．また，手法の有効性を調べるため，被験者にタスクを与え，被験者が実装したソースコードと，ツールが提示したコード片を比較するという実験を行った．被験者は提案手法を実装したツールを含む 3 つのツールを用いてタスクを完成させた．実験の結果，提案手法により既存手法と比べて効率的な再利用支援ができることを確認した．今後の課題としては，提案手法をウェブシステムとして実装し，多くの人に使用してもらえる環境を構築する，提示するコード片の範囲を調整する際に，プログラム構造だけでなく変数の依存関係を考慮する，などを考えている．

謝辞 本論文は，日本学術振興会科学研究費補助金基盤研究 (S)(課題番号：25220003)，挑戦的萌芽研究 (課題番号：24650011)，及び文部科学省科学研究費補助金若手研究 (A)(課題番号：24680002) の支援を受けて行われた．

文 献

[1] K. Inoue, R. Yokomori, T. Yamamoto, M. Matsushita, and S. Kusumoto. Ranking significance of software components based on use relations. *IEEE Transactions Software Engineering*, Vol. 31, No. 3, pp. 213–225, 2005.

[2] M. Grechanik, C. Fu, Q. Xie, C. McMillan, D. Poshyvanyk, and C. Cumby. A search engine for finding highly relevant applications. In *Proc. of the 32nd ACM/IEEE International Conference on Software Engineering*, pp. 475–484, 2010.

[3] C. McMillan, M. Grechanik, D. Poshyvanyk, Q. Xie, and C. Fu. Portfolio: Finding relevant functions and their usage. In *Proc. of the 33rd International Conference on Software Engineering*, pp. 111–120, 2011.

[4] T. Ishihara, K. Hotta, Y. Higo, and S. Kusumoto. Reusing reused code. In *Proc. of the 20th Working Conference on Reverse Engineering*, pp. 457–461, 10 2013.

[5] Java development tools. http://eclipse.org/jdt/.

[6] K. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, Vol. 28, No. 1, pp. 11–21, 1972.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, Vol. 30, No. 1-7, pp. 107–117, Apr. 1998.

[8] A. M. Collins and E.F. Loftus. A spreading-activation theory of semantic processing. *Psychological Review*, Vol. 82, No. 6, pp. 407–428, 1975.

[9] F. Crestani. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, Vol. 11, pp. 453–482, 1997.